Object Detection Networks on Convolutional Feature Maps

Shaoqing Ren, Kaiming He, Ross Girshick, Xiangyu Zhang, and Jian Sun

Abstract—Most object detectors contain two important components: a feature extractor and an object classifier. The feature extractor has rapidly evolved with significant research efforts leading to better deep convolutional architectures. The object classifier, however, has not received much attention and many recent systems (like SPPnet and Fast/Faster R-CNN) use simple multi-layer perceptrons. This paper demonstrates that carefully designing deep networks for object classification is just as important. We experiment with region-wise classifier networks that use shared, region-independent convolutional features. We call them "Networks on Convolutional feature maps" (NoCs). We discover that aside from deep feature maps, a *deep* and *convolutional* per-region classifier is of particular importance for object detection, whereas latest superior image classification models (such as ResNets and GoogLeNets) do not directly lead to good detection accuracy without using such a per-region classifier. We show by experiments that despite the effective ResNets and Faster R-CNN systems, the design of NoCs is an essential element for the 1st-place winning entries in ImageNet and MS COCO challenges 2015.

1 Introduction

Most object detectors contain two important components: a feature extractor and an object classifier. The feature extractor in traditional object detection methods is a hand-engineered module, such as HOG [1]. The classifier is often a linear SVM (possibly with a latent structure over the features) [2], a non-linear boosted classifier [3], or an additive kernel SVM [4].

Large performance improvements have been realized by training deep ConvNets [5] for object detection. R-CNN [6], one particularly successful approach, starts with a pre-trained ImageNet [7] classification network and then fine-tunes the ConvNet, end-to-end, for detection. Although the distinction between the feature extractor and the classifier becomes blurry, a logical division can still be imposed. For example, an R-CNN can be thought of as a convolutional feature extractor, ending at the last pooling layer, followed by a multi-layer perceptron (MLP) classifier. This methodology, however, appears rather different from traditional methods.

A research stream [8], [9], [10], [11] attempting to bridge the gap between traditional detectors and deep ConvNets creates a hybrid of the two: the feature extractor is "upgraded" to a pre-trained deep ConvNet, but the classifier is left as a traditional model, such as a DPM [8], [9], [10] or a boosted classifier [11]. These hybrid approaches outperform their HOG-based counterparts [2], [3], but still lag far behind R-CNN, even when the hybrid model is trained end-to-end [10]. Interestingly, the detection accuracy of these hybrid methods is close to that of R-CNN when using a linear SVM on the last convolutional features, without using the multiple fully-connected layers[1].

The SPPnet approach [12] for object detection occupies a middle ground between the hybrid models and R-CNN. SPPnet, like the hybrid models but *unlike* R-CNN, uses convolutional layers to extract full-image features. These convolutional features are independent of region proposals and are shared by all regions, analogous to HOG features. For classification, SPPnet uses a region-wise MLP, just like R-CNN but *unlike* hybrid methods. SPPnet is further developed in the latest detection systems including Fast R-CNN [13] and Faster R-CNN [14], which outperform the hybrid methods.

From these systems [12], [13], [14], a prevalent strategy for object detection is now: use convolutional layers to extract *region-independent* features, followed by *region-wise* MLPs for classification. This strategy was, however, historically driven by pre-trained classification architectures similar to AlexNet [5] and VGG nets [15] that end with MLP classifiers.

In this paper, we provide an in-depth investigation into object detection systems from the perspective of classifiers aside from features. We focus on *region-wise* classifier architectures that are on top of the shared, region-independent convolutional features. We call them "Networks on Convolutional feature maps", or *NoCs* for short. Our study brings in new insights for understanding the object detection systems.

Our key observation is that carefully designed region-wise classifiers improve detection accuracy over what is typically used (MLPs). We study three NoC families: MLPs of various depths, ConvNets of various depths, and ConvNets with maxout [16] for latent scale selection, where the latter two are unex-

- *The majority of this work was done when the authors were with Microsoft Research.*
- *S. Ren is with University of Science and Technology of China.*
- *K. He and R. Girshick are with Facebook AI Research.*
- *X. Zhang is with Xi'an Jiaotong University.*
- *J. Sun is with Megvii.*

1. The mAP on PASCAL VOC 2007 is 45-47% [8], [9], [10], [11] for hybrid methods, and is 47% for R-CNN that just uses SVM on the last convolutional layer. Numbers are based on AlexNet [5].

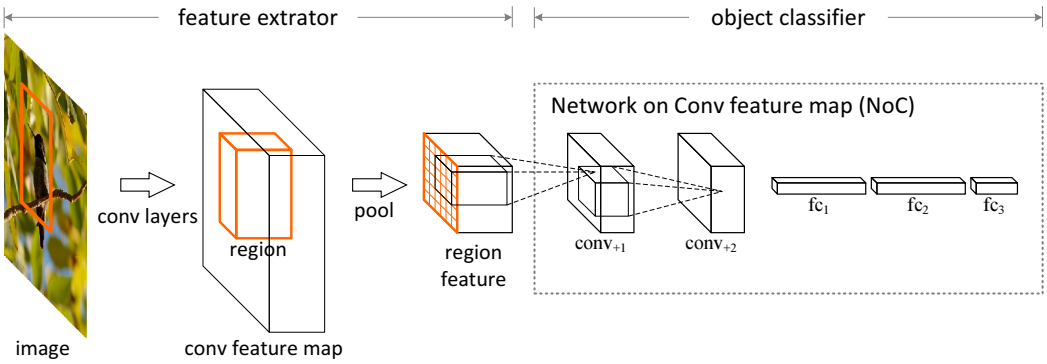

Figure 1: Overview of NoC. The convolutional feature maps are generated by the shared convolutional layers. A feature map region is extracted and RoI-pooled into a fixed-resolution feature. A new network, called a NoC, is then designed and trained on these features. In this illustration, the NoC architecture consists of two convolutional layers and three fully-connected layers.

plored families in previous works [12], [13], [14]. Ablation experiments suggest that: (i) a *deep* region-wise classifier is important for object detection accuracy, in addition to deep shared features; (ii) *convolutional* layers for extracting *region-wise* features are effective, and are complementary to the effects for extracting full-image shared features.

Based on these observations, we present an effective way of plugging "fully convolutional" image classifiers (such as ResNets [17] and GoogLeNets [18]) into the Faster R-CNN [14] system that was designed for the "semi-convolutional" VGG nets [15]. We report that *superior image classification backbones* (e.g., *ResNets and GoogLeNets) do not directly lead to better object detection accuracy*, and a *deep*, *convolutional* NoC is an essential element for outstanding detection performance, in addition to Faster R-CNN and extremely deep ResNets (more details in Table 8).

In summary, through NoC we investigate the region-wise classifiers from different aspects, which are orthogonal to the investigation of features. We believe the observations in this paper will improve the understandings of ConvNets for object detection and also boost the accuracy of prevalent detectors such as Faster R-CNN [14].

2 Related Work

Traditional Object Detection. Research on object detection in general focuses on both features and classifiers. The pioneering work of Viola and Jones [19] uses simple Haar-like features and boosted classifiers on sliding windows. The pedestrian detection method in [1] proposes HOG features used with linear SVMs. The DPM method [2] develops deformable graphical models and latent SVM as a sliding-window classifier. The Selective Search paper [4] relies on spatial pyramid features [20] on dense SIFT vectors [21] and an additive kernel SVM. The Regionlet method [3] learns boosted classifiers on HOG and other features.

ConvNet-based Object Detection. Convolutional layers can be applied to images of arbitrary size yielding proportionally-sized feature maps. In the Overfeat method [22], the fully-connected layers are used on each sliding window of the convolutional feature maps for efficient classification, localization, and detection. In the SPP-based object detection method [12], features are pooled from proposal regions [4] on convolutional feature maps, and fed into the original fully-connected layers for classifying.

Concurrent with this work, several papers [13], [14], [23], [24] improve on the SPPnet method, inheriting the same logical division of shared convolutional features and region-wise MLP classifiers. In Fast R-CNN [13], the shared convolutional layers are fine-tuned end-to-end through Region-of-Interest pooling layers. In Faster R-CNN [14], the shared features are also used for proposing regions and reducing the heavy proposal burdens. The "R-CNN minus R" method [23] waives the requirement of region proposal by using pre-defined regions in the SPPnet system. In the Multi-Region method [24], the features are pooled from regions of multiple sizes to train an ensemble of models.

Despite the improvements, these systems [12], [13], [14], [23], [24] all use MLPs as region-wise classifiers. This logical division naturally applies to a series of networks, such as AlexNet [5], Zeiler and Fergus's (ZF) net [25], OverFeat [22], and VGG nets [15], which all have multiple fine-tunable fc layers. But this is not the case for fully convolutional classification networks, *e.g.*, ResNet [17] and GoogleNet [18], that have *no hidden fully-connected (fc) layers*. We show that it is nontrivial for Fast/Faster R-CNN to achieve good accuracy using this type of networks.

3 Ablation Experiments

Firstly we present carefully designed ablation experiments on the PASCAL VOC dataset [26]. We note that experiments in this section are mainly designed based on the SPPnet system. Particularly, in this section we consider the following settings: (i) the shared feature maps are frozen (which are fine-tunable with Fast R-CNN [13]) so we can focus on the classifiers; (ii) the proposals are pre-computed from Selective Search

[4] (which can be replaced by a Region Proposal Network (RPN) [14]), and (iii) the training step ends with a post-hoc SVM (in contrast to the end-to-end softmax classifier in Fast R-CNN [13]). We remark that observations in this section are in general valid when these restricted conditions are relaxed or removed [13], [14], as shown in the next section with Faster R-CNN [14].

Experimental Settings

We experiment on the PASCAL VOC 2007 set [26]. This dataset covers 20 object categories, and performance is measured by mAP on the *test* set of 5k images. We investigate two sets of training images: (i) the original *trainval* set of 5k images in VOC 2007, and (ii) an augmented set of 16k images that consists of VOC 2007 trainval images and VOC 2012 trainval images, following [27].

As a common practice [6], [12], we adopt deep CNNs pre-trained on the 1000-class ImageNet dataset [7] as feature extractors. In this section we investigate Zeiler and Fergus's (ZF) model [25] and VGG models [15]. The ZF model has five convolutional (conv) layers and three fully-connected (fc) layers. We use a ZF model released by [12][2]. The VGG-16/19 models have 13/16 conv layers and three fc layers, released by [15][3].

Outline of Method

We apply the conv layers of a pre-trained model to compute the convolutional feature map of the entire image. As in [12], we extract feature maps from multiple image scales. In this section these pre-trained conv layers are frozen and not further tuned as in [12], so we can focus on the effects of NoCs.

We extract ~2,000 region proposals by Selective Search [4]. We pool region-wise features from the shared conv feature maps using Region-of-Interest (RoI) pooling [13], [12]. RoI pooling produces a *fixed-resolution* ($m \times m$) feature map for each region, in place of the last pooling layer in the pre-trained model (6×6 for ZF net and 7×7 for VGG-16/19). The pooled feature map regions can be thought of as tiny multi-channel images (see Fig. 1).

We consider these $m \times m$-sized feature maps as a new data source and design various NoC architectures to classify these data. The NoC structures have multiple layers, and the last layer is an (n+1)-way classifier for n object categories plus background, implemented by an (n+1)-d fc layer followed by softmax. Each NoC is trained by backpropagation and stochastic gradient descent (SGD). After network training, we use the second-to-last fc layer in the NoC to extract features from regions, and train a linear SVM classifier for each category using these features, for a fair comparison with [6], [12]. The implementation details follow those in [12].

For inference, the RoI-pooled features are fed into the NoC till the second-to-last fc layer. The SVM classifier is then used to score each region, followed by non-maximum suppression [6].

Next we design and investigate various NoC architectures as classifiers on the RoI-pooled features.

2. https://github.com/ShaoqingRen/SPP_net/
3. www.robots.ox.ac.uk/~vgg/research/very_deep/

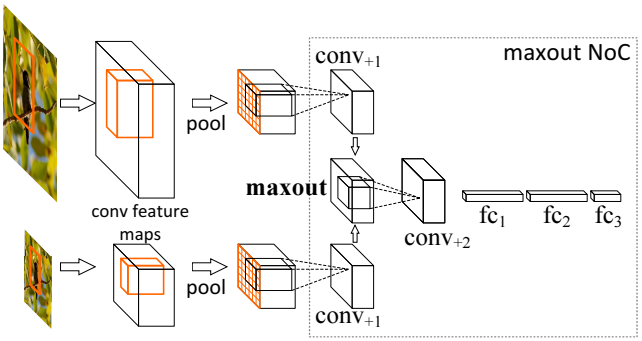

Figure 2: A maxout NoC of "c256-**mo**-c256-f4096-f4096-f21". The features are RoI-pooled from two feature maps computed at two scales. In this figure, maxout is used after conv_{+1}.

3.1 Using MLP as NoC

A simple design of NoC is to use fc layers only, known as a multi-layer perceptron (MLP) [28]. We investigate using 2 to 4 fc layers. The last fc layer is always (n+1)-d with softmax, and the other fc layers are 4,096-d (with ReLU [29]). For example, we denote the NoC structure with 3 fc layers as "f4096-f4096-f21" where 21 is for the VOC categories (plus background).

Table 1 shows the results of using MLP as NoC. Here we *randomly initialize* the weights by Gaussian distributions. The accuracy of NoC with 2 to 4 fc layers increases with the depth. Compared with the SVM classifier trained on the RoI features ("SVM on RoI", equivalent to a 1-fc structure), the 4-fc NoC as a classifier on the same features has 7.8% higher mAP. Note that in this comparison the NoC classifiers have *no pre-training* (randomly initialized). The gain is solely because that MLPs are better classifiers than single-layer SVMs. In the special case of 3 fc layers, the NoC becomes a structure similar to the region-wise classifiers popularly used in SPPnet [12] and Fast/Faster R-CNN [13], [14].

3.2 Using ConvNet as NoC

In recent detection systems [12], [13], [14], [23], [24], conv layers in the pre-trained models are thought of as *region-independent* feature extractors, and thus are shared on the entire image *without being aware of the regions that are of interest*. Although this is a computationally efficient solution, it misses the opportunities of using conv layers to learn *region-aware* features that are fit to the regions of interest (instead of full images). We investigate this issue from the NoC perspective, where the NoC classifiers may have their own conv layers.

method	architecture	VOC 07
SVM on RoI	f21	45.8
2fc NoC	f4096-f21	49.0
3fc NoC	f4096-f4096-f21	53.1
4fc NoC	f4096-f4096-f4096-f21	**53.6**

Table 1: Detection mAP (%) of **NoC as MLP** for PASCAL VOC 07 using a ZF net. The training set is PASCAL VOC 07 trainval. The NoCs are randomly initialized. No bbox regression is used.

method	architecture	VOC 07	07+12
3fc NoC	f4096-f4096-f21	53.1	56.5
1conv3fc NoC	c256-f4096-f4096-f21	**53.3**	58.5
2conv3fc NoC	c256-c256-f4096-f4096-f21	51.4	**58.9**
3conv3fc NoC	c256-c256-c256-f4096-f4096-f21	51.3	58.8

Table 2: Detection mAP (%) of **NoC as ConvNet** for PASCAL VOC 07 using a ZF net. The training sets are PASCAL VOC 07 trainval and 07+12 trainval respectively. The NoCs are randomly initialized. No bbox regression is used.

method	architecture	VOC 07+12
2conv3fc NoC	c256-c256-f4096-f4096-f21	58.9
mo input	**mo**-c256-c256-f4096-f4096-f21	60.1
mo conv_{+1}	c256-**mo**-c256-f4096-f4096-f21	**60.7**
mo fc_1	c256-c256-f4096-**mo**-f4096-f21	60.3
mo output	c256-c256-f4096-f4096-f21-**mo**	60.1

Table 3: Detection mAP (%) of **maxout NoC** for PASCAL VOC 07 using a ZF net. The training set is 07+12 trainval. The NoCs are randomly initialized. No bbox regression is used.

We investigate using 1 to 3 additional conv layers (with ReLU) in a NoC. We use 256 conv filters for the ZF net and 512 for the VGG net. The conv filters have a spatial size of 3×3 and a padding of 1, so the $m \times m$ spatial resolution is unchanged. After the last additional conv layer, we apply three fc layers as in the above MLP case. For example, we denote a NoC with 2 conv layers as "c256-c256-f4096-f4096-f21".

In Table 2 we compare the cases of no conv layer (3-layer MLP) and using 1 to 3 additional conv layers. Here we still randomly initialize all NoC layers. When using VOC 07 trainval for training, the mAP is nearly unchanged when using 1 additional conv layer, but drops when using more conv layers. We observe that the degradation is a result of overfitting. The VOC 07 trainval set is too small to train deeper models. However, NoCs with conv layers show improvements when trained on the VOC 07+12 trainval set (Table 2). For this training set, the 3fc NoC baseline is lifted to 56.5% mAP. The advanced 2conv3fc NoC improves over this baseline to 58.9%. This justifies the effects of the additional conv layers. Table 2 also shows that the mAP gets saturated when using 3 additional conv layers.

Using a ConvNet as a NoC is not only effective for the ZF and VGG nets. In fact, as we show in the next section (Table 8), this design is of central importance for Faster R-CNN using ResNets [17] and other fully convolutional pre-trained architectures.

method	model	init.	VOC 07	07+12
SVM on RoI	ZF	-	45.8	47.7
3fc NoC	ZF	random	53.1	56.5
		pre-trained	**55.8**	**58.0**
maxout 2conv3fc NoC	ZF	random	54.7	60.7
		pre-trained	**57.7**	**62.9**
maxout 2conv3fc NoC	VGG-16	random	59.4	65.0
		pre-trained	**63.3**	**68.8**

Table 4: Detection mAP (%) of NoC for PASCAL VOC 07 using ZF/VGG-16 nets with different initialization. The training sets are PASCAL VOC 07 trainval and PASCAL VOC 07+12 trainval respectively. No bounding box regression is used.

3.3 Maxout for Scale Selection

Our convolutional feature maps are extracted from multiple discrete scales, known as a feature pyramid [2]. In the above, a region feature is pooled from a single scale selected from the pyramid following [12]. Next, we incorporate a local competition operation (maxout) [16] into NoCs to improve scale selection from the feature pyramid.

To improve scale invariance, for each proposal region we select two adjacent scales in the feature pyramid. Two fixed-resolution ($m \times m$) features are RoI-pooled, and the NoC model has two data sources. Maxout [16] (element-wise max) is a widely considered operation for merging two or multiple competing sources. We investigate NoCs with maxout used after different layers. For example, the NoC model of "c256-mo-c256-f4096-f4096-f21" is illustrated in Fig. 2. When the maxout operation is used, the two feature maps (for the two scales) are merged into a single feature of the same dimensionality using element-wise max. There are two pathways before the maxout, and we let the corresponding layers in both pathways share their weights. Thus the total number of weights is unchanged when using maxout.

Table 3 shows the mAP of the four variants of maxout NoCs. Their mAP is higher than that of the non-maxout counterpart, by up to 1.8% mAP. We note that the gains are observed for all variants of using maxout, while the differences among these variants are marginal.

3.4 Fine-tuning NoC

In the above, all NoC architectures are *initialized randomly*. Whenever possible, we can still transfer weights from a pre-trained architecture and fine-tune the NoCs. The comparison of random initialization *vs*. fine-tuning provides new insights into the impacts of the well established fine-tuning strategy [6].

For the fine-tuning version, we initialize the two 4096-d layers by the two corresponding fc layers in the pre-trained model. As such, the fine-tuned 3-fc NoC becomes equivalent to the SPPnet object detection system [12]. For the cases of additional conv layers, each conv layer is initialized to the identity mapping,

	NoC	depth (feature)	depth (classifier)	depth (total)	mAP (%)
VGG-16	3fc	13	3	16	64.6
VGG-19	3fc	16	3	19	65.1
VGG-16	2conv3fc	13	5	18	**66.1**
VGG-16	maxout 2conv3fc	13	5	18	**68.8**

Table 5: Detection results for PASCAL VOC 07 using VGG nets. The training set is PASCAL VOC 07+12 trainval. The NoC is the fine-tuned version (Sec. 3.4). No bounding box regression is used.

and thus the initial network state is equivalent to the pre-trained 3fc structure. We compare the results of an SVM on RoI, randomly initialized NoC, and fine-tuned NoC initialized in the above way. Table 4 shows the cases of two NoCs.

Unsurprisingly, the fine-tuned models boost the results. However, it is less expected to see that the randomly initialized NoCs produce excellent results. Compared with the SVM counterpart using the same RoI-pooled features (47.7%, Table 4), the randomly initialized NoC (60.7%) showcases an improvement of 13.0%, whereas the fine-tuned counterpart (62.9%) has an extra 2.2% gain. This indicates that the fine-tuning procedure, for the classifier, can obtain a majority of accuracy via training a *deep* network on the detection data.

3.5 Deep Features *vs.* Deep Classifiers

We further show by experiments that a deep classifier has *complementary* effects to deep features. Table 5 shows the NoC results using the VGG models [15]. The mAP of the baseline 3fc NoC is 64.6% with VGG-16. With the network replaced by the deeper VGG-19, the depth of shared features is increased by 3, and the mAP is increased by 0.5% to 65.1%. On the other hand, when the depth of region-aware classifier is increased (but still using the VGG-16 features), the mAP is increased by 1.5% to 66.1%. This means that for exploiting very deep networks, the depth of features and the depth of classifiers are both important.

3.6 Error Analysis

Our best NoC using VGG-16 has 68.8% mAP (Table 5). To separately investigate the gains that are caused by features (stronger pre-trained nets) and classifiers (stronger NoCs), in Fig. 3 we analyze the errors of using two sets of pre-trained features (ZF *vs*. VGG-16) and two NoCs (3fc *vs*. maxout 2conv3fc). We use the diagnosis tool of [30].

The errors can be roughly decomposed into two parts: *localization* error and *recognition* error. Localization error ("Loc") is defined [30] as the false positives that are correctly categorized but have no sufficient overlapping with ground truth. Recognition error involves confusions with a similar category ("Sim"), confusions with a dissimilar category ("Oth"), and confusions with background ("BG").

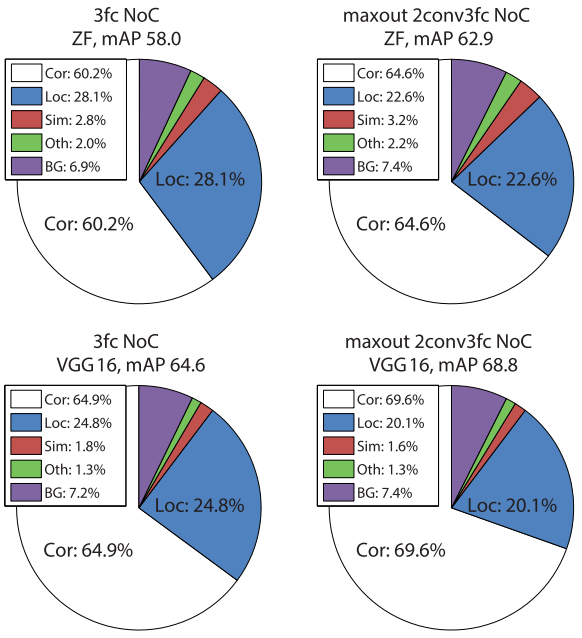

Figure 3: Distribution of top-ranked True Positives (TP) and False Positives (FP), generated by the published diagnosis code of [30]. The types of positive predictions are categorized [30] as Cor (correct), Loc (false due to poor localization), Sim (confusion with a similar category), Oth (confusion with a dissimilar category), BG (fired on background). The total number of samples in each disk is the same and equal to the total number of ground-truth labels [30]. More explanations are in the main text.

Fig. 3 shows that VGG-16 in general has lower *recognition* error than the ZF net, when using the same classifiers (*e.g.*, 1.6%+1.3%+7.4% *vs*. 3.2%+2.2%+7.4%). This suggests that the region-independent features perform more prominent for *recognizing object categories*. On the other hand, when using a stronger NoC (maxout 2conv3fc), *the localization error* is substantially reduced compared with the 3fc baseline (22.6% *vs*. 28.1% with ZF, and 20.1% *vs*. 24.8% with VGG-16). This suggests that the NoCs mainly account for localizing objects. This can be explained by the fact that localization-sensitive information is only extracted after RoI pooling and is used by NoCs.

3.7 Comparisons of Results

In Table 6 and Table 7, we provide system comparisons with recent state-of-the-art results, including R-CNN [6], SPPnet [12], and the latest Fast/Faster R-CNN [13], [14] that are contemporary to this work. We note that all methods in Table 6 and Table 7 are based on Selective Search (SS) proposals [4] (∼2,000 regions per image), except for Faster R-CNN [14] that uses learned proposals.

Our method achieves 71.6% mAP on the PASCAL VOC 2007 test set. This accuracy is higher than Fast R-CNN [13] that also uses SS proposals, and lower than Faster R-CNN [14] that uses learned proposals.

method	training data	mAP (%)
R-CNN [6]	07	62.2
R-CNN [6] + bb	07	66.0
SPPnet [12]	07	60.4
SPPnet [12]	07+12	64.6
Fast R-CNN [13]	07+12	70.0
Faster R-CNN [14]	07+12	**73.2**
NoC [ours]	07+12	68.8
NoC [ours] + bb	07+12	**71.6**

Table 6: Detection results for the PASCAL VOC 2007 test set using the VGG-16 model [15]. Here "bb" denotes post-hoc bounding box regression [6].

method	training data	mAP (%)
R-CNN [6]	12	59.2
R-CNN [6] + bb	12	62.4
Fast R-CNN [13]	07++12	68.4
Faster R-CNN [14]	07++12	**70.4**
NoC [ours]	07+12	67.6
NoC [ours] + bb	07+12	**68.8**

Table 7: Detection results for the PASCAL VOC 2012 test set using the VGG-16 model [15]. Here "bb" denotes post-hoc bounding box regression [6].

Nevertheless, Fast/Faster R-CNN [13], [14] essentially applies a 3-fc NoC structure as the region-wise classifier, and thus the effect of NoCs is orthogonal to theirs. This effect is particularly prominent using the ResNets [17] as we show in the next section.

3.8 Summary of Observations

The following key observations can be concluded from the above subsections:

(i) A **deeper** region-wise classifier is useful and is in general orthogonal to deeper feature maps.

(ii) A **convolutional** region-wise classifier is more effective than an MLP-based region-wise classifier.

These observations are strongly supported by the experimental results on the more challenging MS COCO dataset (Table 8), as we introduced in the next section.

4 NoC for Faster R-CNN with ResNet

The Fast/Faster R-CNN systems [13], [14] have shown competitive accuracy and speed using VGG nets. For networks similar to ZF and VGG-16, Fast/Faster R-CNN are naturally applicable and their region-wise classifiers are 3fc NoCs. However, for "fully convolutional" models such as GoogleNets [18] and ResNets [17], there are no hidden fc layers for building region-wise classifiers. *We demonstrate that the NoC design is an essential factor for Faster R-CNN [14] to achieve superior results using ResNets*.

Experimental Settings

In this section we experiment on the more challenging MS COCO dataset [31] with 80 categories. We train the models on the 80k train set, and evaluate on the 40k val set. We evaluate both COCO-style AP (@ IoU $\in$ [0.5, 0.95]) as well as AP@0.5 and AP@0.75. We adopt the same hyper-parameters as in [17] for training Faster R-CNN on MS COCO.

We compare network architectures of VGG-16 [15], GoogleNet [18], and ResNet-101 [17]. The VGG-16 has center crop top-1 error of 28.5% on the ImageNet classification val set. Regarding GoogleNet, we train the BN-Inception model [32] on ImageNet classification. Our reproduced GoogleNet has center crop top-1 error of 26.4%, close to that reported in [32] (25.2%). The 101-layer ResNet is released by the authors of [17], with center crop top-1 error of 23.6%. Both GoogleNet and ResNet have no hidden fc layer, and instead end with global average pooling and a 1000-d classifier.

Unlike the above section that is based on the SPPnet framework, in this section we use the more advanced Faster R-CNN [14] detector. The main differences are: (i) the entire networks including the features are fine-tuned end-to-end [13]; (ii) the proposals are learned by a RPN [14] with features shared; (iii) instead of post-hoc SVM, a softmax classifier and a jointly learned bounding box regressor [13] are learned end-to-end. Nevertheless, these differences do not affect the design of the NoCs.

Experimental Results

Table 8 shows the results on MS COCO val. We discuss by diving the results into 3 cases as following.

Naïve Faster R-CNN. By this we mean that the RoI pooling layer is naïvely adopted after the last convolutional layer (conv5_3 for VGG-16, inc5b for GoogleNet, and res5c for ResNet). In all cases, we set the output resolution of RoI pooling as 7×7. This is followed by a 81-d classifier (equivalent to a 1fc NoC).

Table 8 shows that VGG-16 has better AP (21.2%) than both GoogleNet (15.2%) and ResNet (16.9%), even though VGG-16 has worse image-level classification accuracy on ImageNet. One reason is that VGG-16 has a stride of 16 pixels on conv5_3, but GoogleNet and ResNet have a stride of 32 pixels on inc5b and res5c respectively. We hypothesize that a finer-resolution feature map (*i.e.*, a smaller stride) contributes positively to object detection accuracy. To verify this, we reduce the stride of GoogleNet/ResNet from 32 to 16 by modifying the last stride=2 operation as stride=1. Then we adopt the "hole algorithm" [33], [34] ("*Algorithme à trous*" [35]) on all following layers to compensate this modification. With a stride of 16 pixels, naïve Faster R-CNN still performs unsatisfactorily, with an AP of 18.6% for GoogleNet and 21.3% for ResNet.

We argue that this is because in the case of naïve Faster R-CNN, VGG-16 has a 3fc NoC but GoogleNet and ResNet has a 1fc NoC (Table 8). As we observed in the above section, a *deeper* region-wise NoC is important, even though GoogleNet and ResNet have deeper feature maps.

Using MLP as NoC. Using the same settings of feature

net	feature	stride	NoC	AP	AP@0.5	AP@0.75
VGG-16	conv5$_3$	16	fc$_{4096}$, fc$_{4096}$, fc$_{81}$	21.2	41.5	19.7
GoogleNet	inc5b	32	fc$_{81}$	15.2	34.7	11.6
GoogleNet	inc5b	32	fc$_{4096}$, fc$_{4096}$, fc$_{81}$	19.8	40.8	17.5
GoogleNet	inc5b, *à trous*	16	fc$_{81}$	18.6	39.4	15.8
GoogleNet	inc5b, *à trous*	16	fc$_{4096}$, fc$_{4096}$, fc$_{81}$	23.6	43.4	23.0
GoogleNet	inc4d	16	inc4e,5a,5b, fc$_{81}$	24.8	44.4	25.2
ResNet-101	res5c	32	fc$_{81}$	16.9	39.6	12.1
ResNet-101	res5c	32	fc$_{4096}$, fc$_{4096}$, fc$_{81}$	21.2	43.1	18.9
ResNet-101	res5c, *à trous*	16	fc$_{81}$	21.3	44.4	18.3
ResNet-101	res5c, *à trous*	16	fc$_{4096}$, fc$_{4096}$, fc$_{81}$	26.3	48.1	25.9
ResNet-101	res4b$_{22}$	16	res5a,5b,5c, fc$_{81}$	**27.2**	**48.4**	**27.6**

Table 8: Detection results of Faster R-CNN on the MS COCO val set. "inc" indicates an inception block, and "res" indicates a residual block.

method	NoC	AP on COCO	mAP on VOC07
res5c, *à trous*	fc$_{n+1}$	21.3	71.9
res5c, *à trous*	fc$_{4096}$, fc$_{4096}$, fc$_{n+1}$	26.3	76.4
res4b$_{22}$	res5a,5b,5c, fc$_{n+1}$	**27.2**	**76.4**

Table 9: Detection results of Faster R-CNN + ResNet-101 on MS COCO val (trained on MS COCO train) and PASCAL VOC 2007 test (trained on 07+12), based on different NoC structures.

maps, we build a deeper MLP NoC by using 3 fc layers (f4096-f4096-fc81). As GoogleNet and ResNet have no pre-trained fc layers available, these layers are randomly initialized which we expect to perform reasonably (Sec. 3.4). This 3fc NoC significantly improves AP by about 4 to 5% for ResNet (21.3% to 26.3% with a stride of 16, and 16.9% to 21.2% with a stride of 32). These comparisons justify the importance of a deeper NoC.

Using ConvNet as NoC. To build a convolutional NoC, we move the RoI pooling layer from the last feature map to an intermediate feature map that has a stride of 16 pixels (inc4d for GoogleNet and res4b$_{22}$ for ResNet). The following convolutional layers (inc4e,5a,5b for GoogleNet and res5a,5b,5c for ResNet) construct the convolutional NoC. The *à trous* trick is not necessary in this case.

With the deeper convolutional NoC, the AP is further improved, *e.g.*, from 26.3% to 27.2% for ResNet. In particular, this NoC greatly improves *localization* accuracy — ResNet's AP@0.75 is increased by 1.7 points (from 25.9% to 27.6%) whereas AP@0.5 is nearly unchanged (from 48.1% to 48.4%). This observation is consistent with that on PASCAL VOC (Fig. 3), where a deep convolutional NoC improves localization.

Table 9 shows the comparisons on PASCAL VOC for Faster R-CNN + ResNet-101. Both MLP and ConvNet as NoC (76.4%) perform considerably better than the 1fc NoC baseline (71.9%), though the benefit of using ConvNet as NoC is diminishing in this case.

Discussions

The above system (27.2% AP and 48.4% AP@0.5) is the foundation of the detection system in the ResNet paper [17]. Combining with orthogonal improvements, the results in [17] secured the 1st place in MS COCO and ImageNet 2015 challenges.

The ablation results in Table 8 indicate that despite the effective Faster R-CNN and ResNet, it is not direct to achieve excellent object detection accuracy. In particular, a *naïve* version of Faster R-CNN using ResNet has low accuracy (21.3% AP), because its region-wise classifier is shallow and not convolutional. On the contrary, a **deep** and **convolutional** NoC is an essential factor for Faster R-CNN + ResNet to perform accurate object detection.

5 Conclusion

In this work, we delve into the detection systems and provide insights about the *region-wise* classifiers. We discover that deep convolutional classifiers are just as important as deep convolutional feature extractors. Based on the observations from the NoC perspective, we present a way of using Faster R-CNN with ResNets, which achieves nontrivial results on challenging datasets including MS COCO.

References

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[2] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *TPAMI*, 2010.

[3] X. Wang, M. Yang, S. Zhu, and Y. Lin, "Regionlets for generic object detection," in *ICCV*, 2013.

[4] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *IJCV*, 2013.

[5] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[8] P.-A. Savalle, S. Tsogkas, G. Papandreou, and I. Kokkinos, "Deformable part models with CNN features," in *Parts and Attributes Workshop, ECCV*, 2014.

[9] R. Girshick, F. Iandola, T. Darrell, and J. Malik, "Deformable part models are convolutional neural networks," in *CVPR*, 2015.

[10] L. Wan, D. Eigen, and R. Fergus, "End-to-end integration of a convolutional network, deformable parts model and non-maximum suppression," in *CVPR*, 2015.

[11] W. Y. Zou, X. Wang, M. Sun, and Y. Lin, "Generic object detection with dense neural patterns and regionlets," in *BMVC*, 2014.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *ECCV*, 2014.

[13] R. Girshick, "Fast R-CNN," in *ICCV*, 2015.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *NIPS*, 2015.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[16] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv:1302.4389*, 2013.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *arXiv prepring arXiv:1506.01497*, 2015.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, and A. Rabinovich, "Going deeper with convolutions," Tech. Rep., 2014. [Online]. Available: http://arxiv.org/pdf/1409.4842v1

[19] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR*, 2001.

[20] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006.

[21] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.

[22] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *ICLR*, 2014.

[23] K. Lenc and A. Vedaldi, "R-cnn minus r," in *BMVC*, 2015.

[24] S. Gidaris and N. Komodakis, "Object detection via a multi-region & semantic segmentation-aware cnn model," in *ICCV*, 2015.

[25] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional neural networks," in *ECCV*, 2014.

[26] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes (VOC) Challenge," *IJCV*, 2010.

[27] P. Agrawal, R. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *ECCV*, 2014.

[28] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, 1989.

[29] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[30] D. Hoiem, Y. Chodpathumwan, and Q. Dai, "Diagnosing error in object detectors," in *ECCV*, 2012.

[31] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," *arXiv:1405.0312*, 2014.

[32] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[33] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[34] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *ICLR*, 2015.

[35] S. Mallat, *A wavelet tour of signal processing*. Academic press, 1999.